Buses

Índice

1. Introducción

Definición

Conjunto de lineas compartidas por distintos elementos de un computador cuya función es permitir la comunicación entre ellos. Los buses no pertenecen a ninguno de los elementos.

Los buses *conectan*:

_ Elementos en la CPU
_ Componentes en una tarjeta (entre tarjetas en un Rack: VME, PCI, Multibus...)
_ Periféricos (IDE, SCASI, Firewire, SUB, SATA, ISA...)

Características

1) ***Tipo de elementos que conecta***: Dedicados o Generales
2) ***Uso***: Buses de tiempo real (CAN-Bus), de sistemas de control, para multiprocesadores (FutureBus) ...
3) ***Ancho***: Existen lineas de Datos, Control y Direcciones. El ancho de la información que se puede transmitir (lineas de datos) lo determina el ancho del bus
4) ***Ancho de Banda***: Capacidad máxima en bytes/s
5) ***Mecánicas***: Dimensiones, contactos ...
6) ***Eléctricas***: Voltajes
7) ***Temporización***: Síncrona o Asíncrona
8) ***Arbitraje***: Centralizado o Distribuido
9) Gestión de las interrupciones

Líneas de Bus

Las lineas de bus se componen de:

_ Datos
_ Direcciones } Multiplexados
_ Control (arbitraje, errores, multiprocesador...)

Buses Multiplexados: Emplea los mismos hilos para enviar información distinta en momentos diferentes. Típicamente Daos y Direcciones. Además se consigue simplificar el diseño.

Clasificación de los Buses

a) ***Por niveles*** (Borrill '81)

_ *Nivel Placa*. Conecta elementos en un chip (VLSI)
N_0: de Chip
N_1: de Tarjeta
_ *Nivel Panel Posterior.* Conecta elementos en una tarjeta
N_2: Conexión entre placas
N_3: Conexión entre componentes
_ *Nivel Interfaz*
N_4: Periféricos (IDE, USB, SATA, Firewire …)
N_5: Bus serie (para conexiones lejanas: RS-232)

Estos niveles forman una jerarquía de buses. Cuanto menor sea la distancia al procesador, más rápido será el bus. Hay distintos buses, con distinta velocidad en el sistema. Para conectar buses de distintas velocidades necesitaremos “bridges” que también realizarán funciones de buffering.

b) Por dedicación

_ *Dedicados*: realizan una única función, son más especializados, más simples y con menor coste. Tienen un mejor rendimiento.
_ *Generales*: bus global más complejo y caro que uno dedicado pero más sencillo y barato que “m” dedicados.

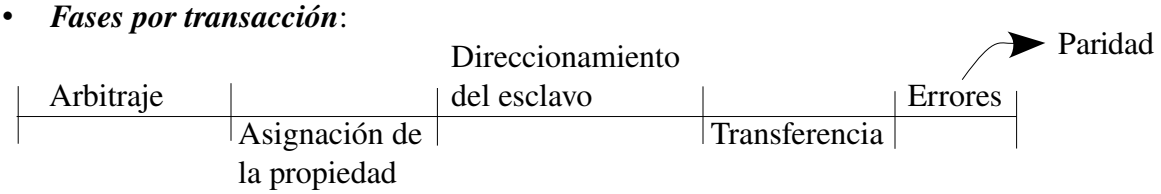
c) Por ocupación

_ *Partición según recurso*: recurso del mismo tipo juntos. Orientado a los procesadores a transferencias elementales tipo CPU – Memoria. Por ejemplo: VME.
_ Partición según bloques funcionales. Orientado a multiprocesadores y a transferencias tipo mensaje. Por ejemplo: FutureBus+, MultibusII.

d) ***Según uso***: Generales
Multiprocesador: FutureBus
Tiempo Real: CAMBUS

Terminología

- ***Operación / Transacción de Bus***: Secuencia completa.
- ***Fuente***: Origen de la información.
  Destino: Donde va a parar la información.
- ***Maestro***: Toma la iniciativa y dirige la operación.
  Esclavo: Sigue al maestro.
- ***Fases por transacción***:



Buses Normalizados

Antes cada fabricante definía sus buses lo cual dificultaba mucho la comunicación entre distintos componentes. Para facilitar la interacción entre componentes de distintos fabricantes los buses se han "normalizado". Siguen un estándar acordado previamente.

2. Transferencia de Datos

Intercambio de señales de control y datos entre el maestro del bus (el que tiene la prioridad en ese momento) y uno o varios esclavos con el objetivo de transferir información.

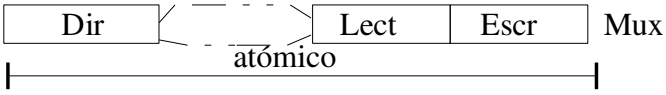
Tipos de Transferencias

a) _ *Mux*: Address/Data

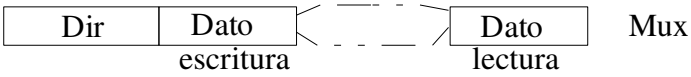
_ *No Mux:*

b) *Lectura – Modificación – Escritura Atómica*. Para implementar cerrojos.

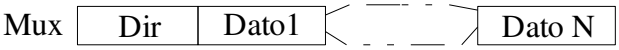
RI ↔ MC (Dir_Mp)
Dir Cerrojo
Ej.: lock.xchg .R1, /Dir_MP
test & set
temp ← M(lock)
M(lock) ← "11111"



c) *Ciclo de Lectura con Verificación* (poco usado)



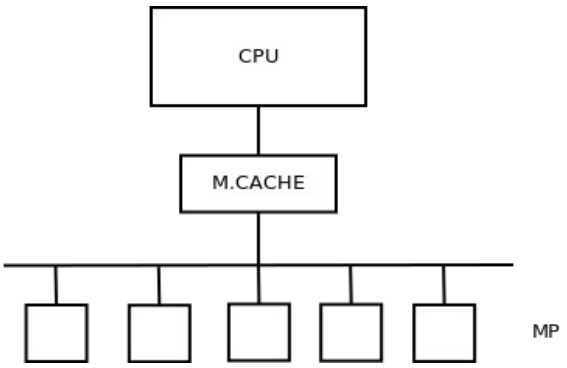
d) *Transferencia de Bloque* (muy usado)



Se emplea para acceso a M.Caché (lectura escritura de una linea de M.Caché simultánea). También se emplea para E/S por DMA por ráfagas.

Los esclavos deben anidar e incrementar la dirección enviada. Hay buses que permiten bloques de tamaño fijo (Nubus: 1, 2, 4, 8 ó 16) y otras que lo permiten variable (VME).

Este esquema suele combinarse con memoria entrelazada simple (misma dirección para todos los módulos) de orden inferior (posiciones consecutivas en módulos consecutivos).



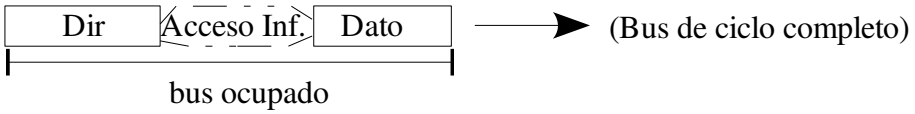
Obs: problema con estas transferencias: Son MUY largas. Puede venir algo urgente y el bus está ocupado y aún queda mucho tiempo para que termine.

Solución: interrumpir o abortar la operación en curso. Existe una línea BCLEAR y volvemos a la fase de arbitraje.

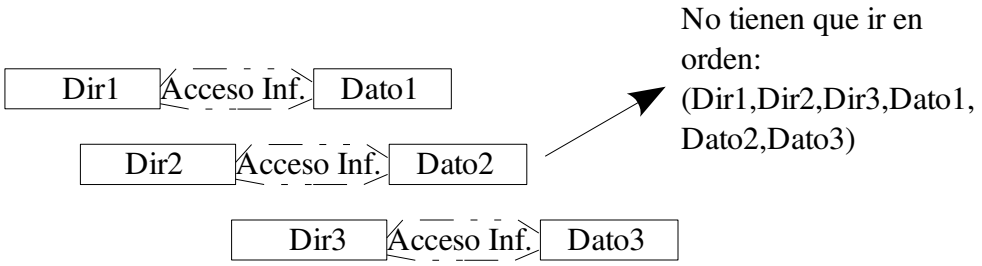
e) Buses de ciclo partido

Mientras se accede ala información (tiempo de acceso) se deja libre el bus para que pueda ser utilizado por otro maestro. Se necesita una mayor complejidad pero aumenta el rendimiento. Orientado a Multiprocesadores.

_ Bus "normal" (MUX) [monoprocesadores]:

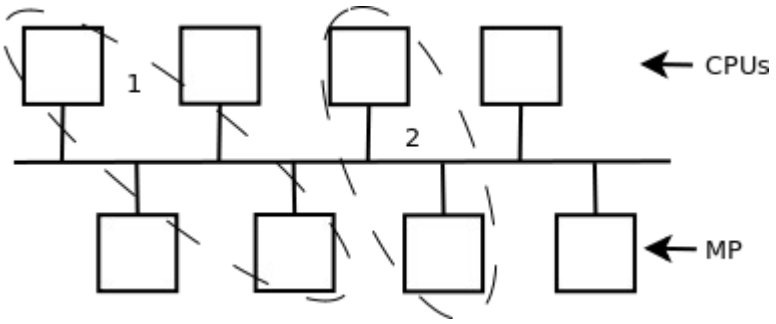


_ Bus de ciclo partido (MUX):



Necesita esclavos más complejos que soliciten el Bus. Además, los esclavos deben almacenar la dirección del maestro que les solicitó el dato. Necesitamos también un lugar donde almacenar datos, ya que puede darse la situación en que el dato esté listo y el bus no esté disponible.

Obs: en este caso viene muy bien la memoria con entrelazado complejo (una dirección distinta para cada módulo) de orden superior (direcciones consecutivas en el mismo módulo)



1.) La CPU pide un dato a MP
2.) Otro módulo pide un dato a otro módulo de MP (si fuese un sólo módulo seria imposible hacer este acceso).
3.) Se resuelve el dato solicitado en 2 y se libera el bus.
4.) Se resuelve el dato solicitado en 1.

Direccionamiento

Cada elemento del sistema tiene una dirección única para dialogar con él sin confusión.

$$Dir\ esclavo \approx Dir\ tarjeta + Dir\ en\ esa\ tarjeta$$

Existen dos posibles direccionamientos:

_ ***Lógico***: Independiente de la posición física de la tarjeta.
Tienen microinterruptores para asignar manualmente la dirección. Este es propenso a errores al ser un sistema manual, por lo que ha entrado en desuso.
_ ***Físico***: Depende de la posición. Durante el arranque del computador se recorren todas las tarjetas y se les asigna una dirección a cada una que guardan en un registro. Se configura de forma automática por el sistema operativo, por lo que no es común que se den errores.

Ej.: PCI, USB, MULTIBUS … Conocido como Plug&Play.

También existe la posibilidad de dialogar con varios esclavos simultáneamente, *direccionamiento múltiple*.

a) ***Broadcall***: lectura múltiple. Ej.: preguntar qué periférico solicitó la interrupción dejando un hilo del bus a cada uno (no se utiliza ya).

b) ***Broadcast & Multicast***: Escritura múltiple. Multicast: Varios. Ej.: protocolos coherencia caché. Broadcast: Todos. Ej.: Reset.

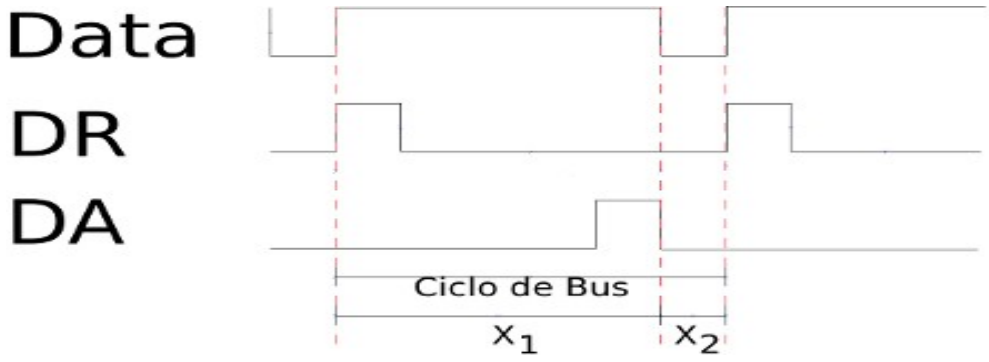
Protocolos de Temporización

Deben sincronizar origen y destino. Tres modos de sincronización:

_ ***Síncrono***, a instantes fijos.
_ ***Asíncrono***, a instantes arbitrarios.
_ ***Semi-síncrono***, mezcla de los anteriores

a) Bus Síncrono

Tenemos dos señales síncronas con el reloj que son DA (Datos Aceptados) generada por destino y DR (Datos Recibidos) generada por origen.



Hay un reloj común y todo sucede en instantes fijos de tiempo marcados por dicho reloj. No hay diálogo entre origen y destino, por lo que no hay confirmación de la operación, pero hay mayor aprovechamiento del bus ya que en cada ciclo leemos algo.

b) Bus Asíncrono

Hay diálogo entre fuente y destino, las acciones se realizan en respuesta a este diálogo y al reloj. Tenemos tres posibilidades:

i. Sin interbloqueos

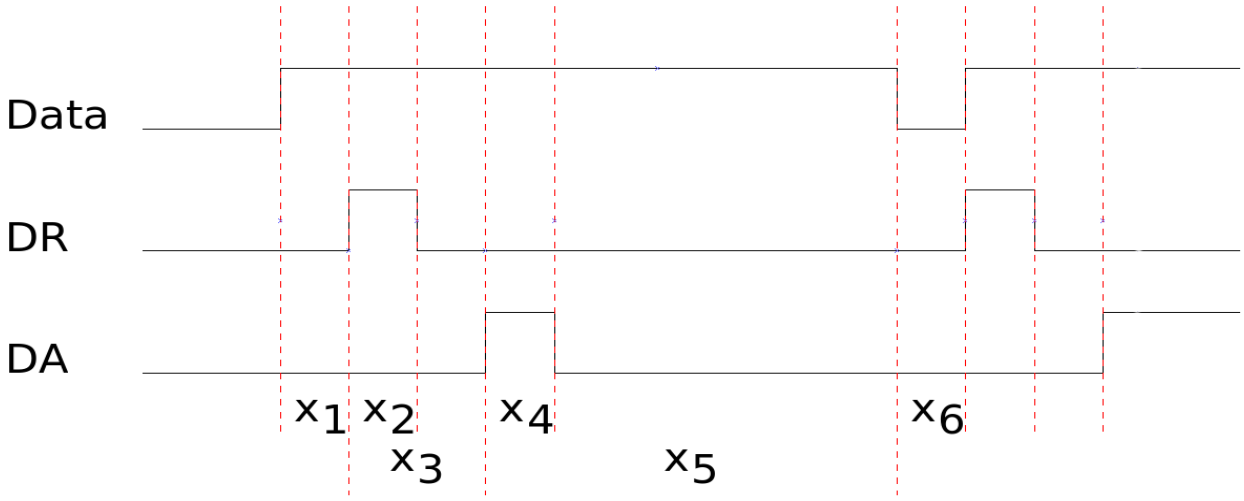
Duración fija de t_2 y t_4, también de t_1 y t_6, pero no es relevante. t_3 y t_5 son variables. t_3 marca el tiempo de respuesta del destino. t_5 marca el tiempo de respuesta del origen.

ii. Semi-bloqueante

t_2 variable.

iii. Totalmente bloqueante

Duración fija de t_1,t_6 y variable de t_2, t_3, t_4 y t_5



Hay diálogo, por lo que hay cierta verificación. El origen sabe que el destino ha visto al dato. Es algo más lento que el síncrono por las esperas en t_3 y t_5.
Problemas posibles: gran diferencia de velocidad entre origen y destino.
Ejemplos.:

_ ***Fuente muy rápida o destino muy lento***: Empieza el nuevo ciclo con DA aún activo y el origen puede pensar que el destino ya ha leído el dato y quitarlo.

_ ***Destino muy rápido o fuente muy lenta***: cuando el destino desactiva DA ve que DR sigue activo y puede pensar que hay un nuevo dato listo.

c) Bus semi-síncrono

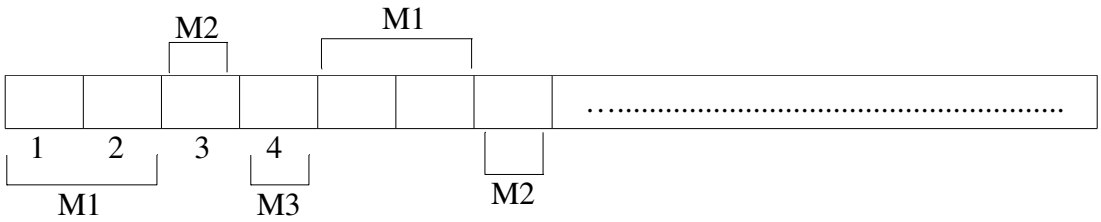
Es como el asíncrono pero al que se le ha incorporado un reloj. Los flancos sólo podrán tener lugar en instantes múltiplo del reloj. Se reduce cuando la posiblidad de perder un flanco por ruido, es menos sensible al ruido pero más lento por lo que hay que esperar a los flancos de reloj. Ej.: PCI.

3. Arbitraje del Bus

Normalmente existe más de un maestro y se hace necesaria una fase de arbitraje previa a la transferencia, para decidir quién es el propietario del bus en cada momento y así evitar posibles conflictos.

Arbitraje Estático

Se hace un reparto previo de la propiedad del bus. Es muy sencillo, pero poco eficaz.
Ej.: Maestro M1: T1,T2 M2: T3 M3: T4



Si un maestro no tiene nada que transmitir se pierde esta ranura del bus.

Arbitraje Dinámico

La propiedad se decide entre los que solicitan usar el bus:

Solicitud → Arbitraje → Asignación de Propiedad

A) Políticas de Gestión del Bus

- ***Por Prioridad*** → Cada maestro tiene una prioridad, y se asigna al más prioritario. Ej.: monoprocesadores, controladores DMA.
- ***Equitativa*** → Busca que ningún maestro monopolice el bus. Ej.: Multiprocesadores entre CPU's
- ***Combinada*** → Algunos maestros por prioridad, y otros siguiendo una política equitativa. Ej.: E/S por DMA en un multiprocesador.

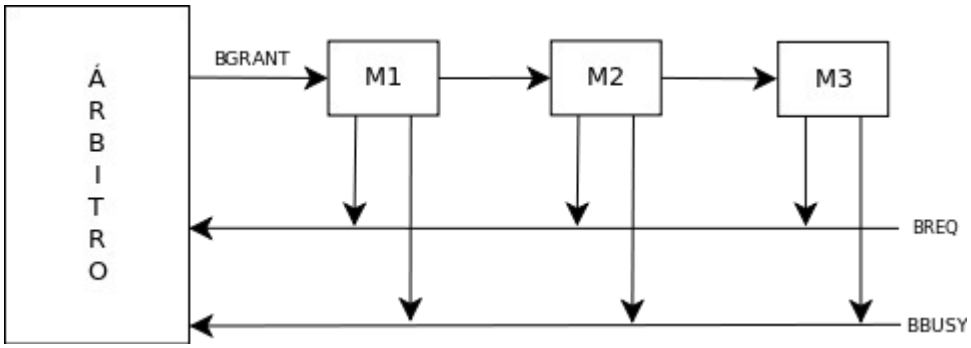
B) Políticas de Gestión del Bus

- ***Bajo Petición*** → Mientras nadie pida el bus, éste pertenece al último maestro que lo pidió.
- ***Al terminar la Transacción*** → Al final de cada transacción vuelve a establecer la prioridad bus.
- ***Expulsivo*** → Se desaloja en mitad de una transferencia si llega un maestro más prioritario que necesita el bus y no puede esperar. Ej.: Transferencia de Bloque.

Mecanismo HW para el arbitraje del Bus

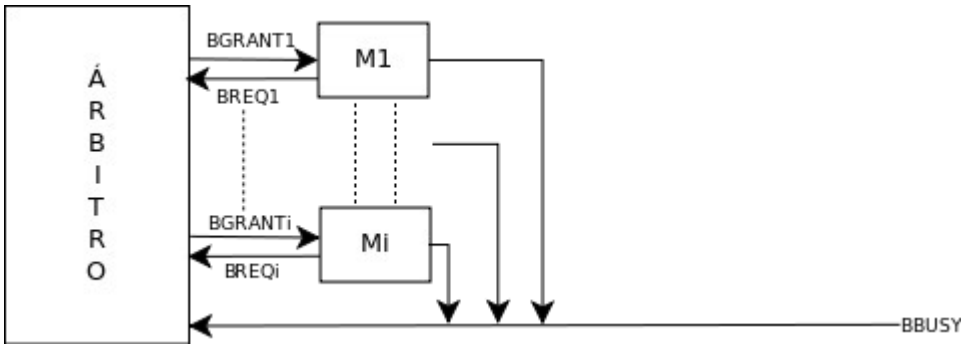
- ***Arbitraje Centralizado*** → sólo existe un único árbitro, al cual todos los maestros le solicitan el uso del bus. El HW es similar al visto en E/S.

  **** Líneas de Petición Compartidas y concesión encadenada (Daisy-Chain)***



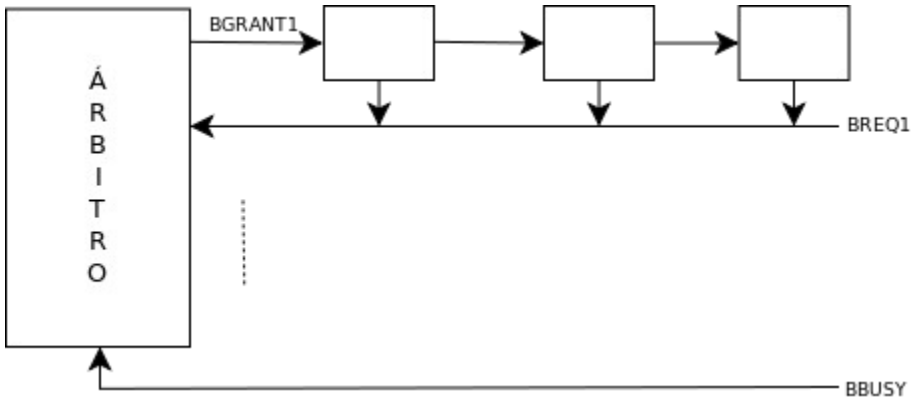
  Este modelo se basa en prioridades (según la posición).
  Por tanto, es un modelo no equitativo, número limitado de maestros y sencillo.

  **** Líneas de Petición y concesión independientes***



  Es más complejo porque hay muchas líneas de control. Es más flexible, admite equidad o prioridad, pero existe un numero limitado de maestros.

**** Combinado***



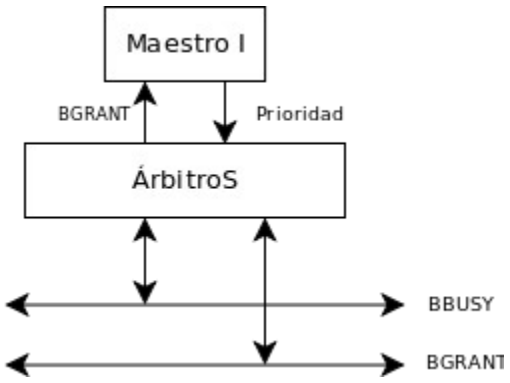
Combina las ventajas de los dos anteriores.

Obs → Puede existir una línea adicional en cualquiera de estos esquemas, para poder implementar políticas expulsivas: Bclear. Este bus va del Árbitro a todos los maestros, y sirve para avisarle si un maestro más prioritario quiere usar el bus.

Obs → Estos esquemas son típicos de monoprocesadores.

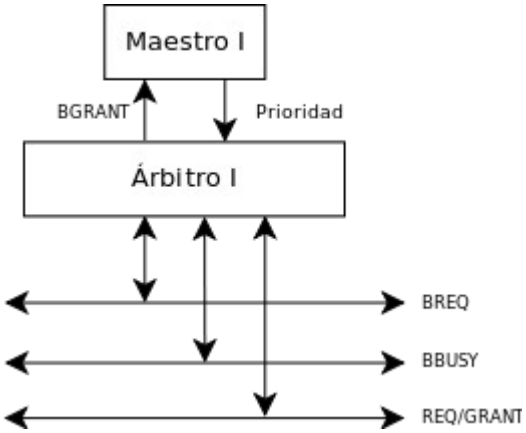
- ***Arbitraje Distribuido*** → Varios árbitros dialogan entre sí, para establecer la propiedad del bus.

**** Prioridades***



El Maestro I pasa su prioridad al Árbitro I, que la vuelca en las líneas de $B_{Req/Grant}$. Luego lee de dichas líneas, si resulta que no es el más prioritario, se retira (quita la prioridad). Cuando al final se han retirado todos menos el más prioritario (sólo quedará u prioridad en las líneas), sabe que el bus le ha sido concedido (los demás se han retirado). Avisa el Árbitro I a su Maestro a través de B_{grant} . Con B_{busy} avisa a los demás cuando terminen.

**** Equidad***



El esquema es muy similar al anterior, se ha añadido una linea B_{Req} . Esta línea es usada por parte de los árbitros menos prioritarios, que se tienen que retirar.

La activan para dejar anotado que les "hubiese gustado usar el bus".

Luego, los maestros más prioritarios que han usado el bus, pueden consultar dicha línea para dejar que los maestros menos prioritarios usen el bus: no volverán a pedir el bus hasta que no se desactive la señal. → Si alguno tiene mucha prisa, ignora esta señal (multiprocesadores).